

Incompressible Navier-Stokes with Particles Software Testing Plan

Applied Numerical Algorithms Group
NERSC Division
Lawrence Berkeley National Laboratory
Berkeley, CA

August 6, 2003

Contents

1	Scope	3
1.1	System Overview	3
2	Reference Documents	4
3	Software Test Environment	5
4	Test Identification	6
4.1	General Information	6
4.1.1	Test Level	6
4.1.2	Test Classes	6
4.2	Planned Tests	6
4.2.1	Test 1 – DiscreteDeltaFn Unit Test	7
4.2.2	Test 2 – DragParticle Unit Test	7
4.2.3	Test 3 – ParticleProjector::computeD Test	7
4.2.4	Test 4 – ParticleProjector::computeInfiniteDomainBCs Test	7
4.2.5	Test 5 – ParticleProjector Test – Single Particle	7
4.2.6	Test 6 – ParticleProjector Test – Multiple Particles	8
4.2.7	Test 7a – AMRINS with particles system – Single Particle Sedimentation	8
4.2.8	Test 7b – AMRINS with particles system – Single Particle coupled with fluid motion	8
4.2.9	Test 8a – AMRINS with particles system – Multiple Particles Sedimentation	8
4.2.10	Test 8b – AMRINS with particles system – effective viscosity test	8
4.2.11	Test 8c – AMRINS with particles system – Multiple particles coupled with fluid motion	8
4.2.12	Test 9 – AMRINS with particles system regression test	9
5	Test Schedules	10
6	Bug Tracking	11

Chapter 1

Scope

The AMR incompressible Navier-Stokes with particles code developed for this project will build heavily on the already-existing AMR incompressible Navier-Stokes (AMRINS) code, which itself relies on the functionality in the Chombo software infrastructure [2], including the `ParticleTools` support for particles in the Chombo library. The software test plan outlined in this document will focus on the additional functionality developed for the particle capability of the code; since AMRINS and Chombo have their own software test plans, it is not necessary to provide for testing the functionality of the pre-existing code. Note, however, that since the software developed for this project uses the AMRINS and Chombo functionality so extensively, changes and bugs in the AMRINS and Chombo code will tend to have effects on the current testing results. The developers for this project are kept abreast of developments in Chombo through CVS notification (which sends e-mail whenever a change is made in the Chombo CVS version-control repositories), and through the ChomboUsers e-mail list.

1.1 System Overview

The software implements an AMR algorithm for solving the incompressible Navier-Stokes equations with forcing due to drag from suspended particles. The algorithm to be used in this work is based on the viscous algorithm presented in the “Incompressible Navier-Stokes Software Design” document, along with the particle projection algorithm presented in the “Incompressible Navier-Stokes with Particles Algorithm Design Document”.

Chapter 2

Reference Documents

In addition to the Chombo design document [2] and the algorithms described in the “Incompressible Navier-Stokes with Particles Algorithm Design Document” [4] and the “Incompressible Navier-Stokes Software Design” document [3] , we will also refer to the “Software Design for Particles in Incompressible Flow” document [5].

Chapter 3

Software Test Environment

The AMRINS for particles software, linked to the AMRINS code and the Chombo software libraries, will be tested. As new functionality is added and current functionality is improved, testing will continue. It is expected that a given time, the AMRINS and particle codes will be in sync with the current state of the Chombo libraries.

This software is primarily intended for use on UNIX/Linux-based systems. In general, the makefiles used in both Chombo and AMRINS require GNU make (gmake). The software itself is designed to be run from a shell, with an inputs file providing run-specific inputs. For data output, the software uses hdf5, so the system must have hdf5-1.4.1 installed. The Chombo and AMRINS software is written in C++ and Fortran77, so working C++ and F77 compilers must be available. We generally use the GNU compiler: both gcc 2.95 and 3.1 have been successfully used to compile this code. In addition, the Chombo Fortran preprocessor uses PERL. If ChomboVis will be used to examine results, then it must be installed as well. ChomboVis additionally requires Python and VTK.

We will test the AMRINS-particles/Chombo combination in a variety of environments, with a variety of compilers. Table 3 lists the platforms and compilers we have successfully compiled and run the AMRINS code:

Testing is done by ANAG personnel, although collaborators have been useful for finding unintended functionality, primarily in the Chombo libraries themselves.

Platform	OS	C++ Compiler	Fortran Compiler
CRAY T3E	unicos	KCC 3.3d	Cray Fortran 3.5.0.4
IBM SP	AIX	KCC 4.0f, xLC 5.0.2.0	IBM XL Fortran 7.1.1.0
Pentium/AMD	Linux	gcc 2.95.3+, Intel C++ 6.0	g77 2.95.3+, PGI Fortran 3.3-2 Intel Fortran 5.0.1
Compaq	OSF	gcc 3.1	Compaq f77 X5.4A-1684-46B5P
Compaq	Linux	gcc 2.95.3	g77 2.95.3
SGI	IRIX	MIPS Pro CC 7.3.1.2m, gcc 2.95.3	MIPS Pro f90 7.3.1.2m

Table 3.1: Platforms and compilers on which the AMRINS code has been tested

Chapter 4

Test Identification

4.1 General Information

In general, the best way to test whether many components are functioning properly is to do a convergence study. For example, for a velocity projection component, a velocity field is initialized on a series of meshes, each a factor of 2 finer than the last. The projection is applied to the velocity field, and then the divergence of the resulting velocity field is computed. If the projection component is properly implemented, the divergence should decrease at second-order rates.

4.1.1 Test Level

In general, most of the testing outlined in this document will be component testing. System-level testing will also be carried out on the entire AMRINS with particles code. It is expected that integration testing is not necessary at this time, because of the small size of the design team.

4.1.2 Test Classes

In general, testing will be structured to evaluate correctness of the code. It is anticipated that since the next phases of code development will be focused on performance enhancement, that performance of the code will be monitored closely, so routine performance testing should be unnecessary, while testing for correctness will be important as changes are made to speed up the code.

4.2 Planned Tests

In this section, we outline the tests planned for the particle code, broken down by functional software unit. All testing codes are written in C++.

4.2.1 Test 1 – DiscreteDeltaFn Unit Test

The discreteDeltaFn class implements the spreading of a point force onto the computational grid. We test this spreading function to ensure that it has been coded correctly.

4.2.2 Test 2 – DragParticle Unit Test

The specific functionality for the DragParticle class will be tested in a series of unit tests. These unit tests will compare the functional results with the analytically computed correct solution.

- a. DragParticle::computeDragForce Test This unit test will ensure that the drag force is computed correctly for a single particle given a prescribed velocity field.
- b. DragParticle::computeK Test This unit test will ensure that the kernel \mathbf{K} is computed correctly for a single particle at a given location \mathbf{x} .
- c. DragParticle::computeProjForce Test This unit test will ensure that the force $f_j K_{ij}$ is computed correctly for a single particle at a given location \mathbf{x} .

4.2.3 Test 3 – ParticleProjector::computeD Test

Unit tests to ensure that the kernel D is being computed correctly by the ParticleProjector class. This will be done in two ways, a test with a single particle, and a test with a group of particles:

- a. Single particle – test function result against analytic solution.
- b. Multiple particles – test for 2nd-order convergence.

4.2.4 Test 4 – ParticleProjector::computeInfiniteDomainBCs Test

Unit test to ensure that the infinite domain boundary condition outlined in Section 3 of [4] is implemented correctly and that it approximates solutions on an infinite domain. Perform an elliptic solve of a reference problem on an infinite domain for which there is an analytic solution (A Gaussian, for example) using the infinite domain boundary condition functionality in the ParticleProjector class. Solution should converge to analytic solution at second-order rates.

4.2.5 Test 5 – ParticleProjector Test – Single Particle

Unit test to ensure that the force due to a single particle is projected correctly. Test for convergence to analytic solution. Also, test to ensure that divergence of resulting field converges to 0 at second-order rates.

4.2.6 Test 6 – ParticleProjector Test – Multiple Particles

Unit test to ensure that force due to multiple particles is projected correctly. Test that resulting field and its divergence converge at second-order rates.

4.2.7 Test 7a – AMRINS with particles system – Single Particle Sedimentation

In this test, a fluid system with a single suspended particle with a gravitational forcing in zero-velocity fluid is tested. The motion of the particle should agree with accepted sedimentation rates from the literature.

4.2.8 Test 7b – AMRINS with particles system – Single Particle coupled with fluid motion

In this test, a fluid system with a single suspended particle is tested. The particle should be advected along with the flow field, while exerting a drag force on the fluid. The solution should converge at second-order in space and time.

4.2.9 Test 8a – AMRINS with particles system – Multiple Particles Sedimentation

A fluid system with multiple particles with a gravitational forcing in a zero-velocity fluid is validated. The motion of the particles should agree with accepted sedimentation rates.

4.2.10 Test 8b – AMRINS with particles system – effective viscosity test

Batchelor [1] presents a solution for the effective viscosity of a dilute suspension of particles. To further validate the code, we will compute the effective viscosity of a dilute suspension of particles using the AMRINS-particles code and compare to the solution derived in [1].

4.2.11 Test 8c – AMRINS with particles system – Multiple particles coupled with fluid motion

In this test, a fluid system with multiple suspended particles in a flow field is tested. The particles should be advected along with the flow, while exerting drag forces on the fluid. The solution should converge at second-order rates in space and time.

4.2.12 Test 9 – AMRINS with particles system regression test

To ensure that Chombo library changes, bug fixes, and related code changes do not cause unintended changes in code results, an AMRINS-particles system regression test will be employed. The AMRINS particles code will be run with a benchmark inputs file and diagnostic quantities are reported at the end of the run. Changes in these diagnostic quantities will indicate changes which will need to be investigated.

Chapter 5

Test Schedules

Once a capability in the code has been verified by the appropriate test, we plan to use these tests as regression tests. We plan to apply the entire test suite once each month to ensure that no unintended changes are introduced, and we also will re-run the test suite after bugs are found and corrected to ensure that new bugs are not introduced.

The system regression test (Test 9) will be done weekly for serial runs, and monthly for the suite of parallel runs, and also after bug fixes and library changes to lessen the possibility of unintended changes in the code.

Also, acceptance tests will be run as stakeholders take possession of the software.

Chapter 6

Bug Tracking

The AMRINS particle code developers (and the Chombo developers) use the ttpro system for bug tracking. When a bug or unexpected behavior in the code is identified, a description is entered in the ANAG ttpro database. As the bug is investigated and fixed, the description is updated and expanded. Once a bug has been fixed, the bug report is “closed” in ttpro, but it remains in the database for future reference if needed. Also, after a bug fix, the regression test (Test # 9) is re-run to ensure that no unanticipated effects have been added.

Chapter 7

Requirements Traceability

The requirements traceability matrix is presented in Figure 7.1. The first column, “Alg Spec No”, connects the entry in matrix with the relevant section of the “Incompressible Navier-Stokes with Particles Design Document” algorithm design document [4]. A parenthetical number refers to a specific equation in [4].

Table 7.1: Requirements Traceability Matrix

Alg Spec No.	Req Statement	S/W module	Test Spec	Test Case #	Verification	Mod. Field
1 (5)	Particle spreading	DiscreteDeltaFn	Particle Test Plan	1	not verified	
1 (2)	Particle class test	DragParticle: computeDragForce	Particle test plan	2a	not verified	
3.1	Particle class test	DragParticle: computeK	Particle test plan	2b	not verified	
4	Particle class test	DragParticle: computeProjForce	Particle test plan	2c	not verified	
4 – 1(a-b)	Particle Projection	ParticleProjector:: computeD	Particle Test Plan	3	not verified	
3	Particle Projection	ParticleProjector:: computeInfiniteDomainBCs	Particle Test Plan	4	not verified	
3	Particle Projection	ParticleProjector:: projectForce	Particle Test Plan	5-6	not verified	
1 (1-2)	AMRINS/ Particle code	AMRINS/particle system	Particle Test Plan	7(ab)-8(ab)	not verified	
1 (1-2)	AMRINS/ Particle code	AMRINS/particle system	Particle Test Plan	9	not verified	

Bibliography

- [1] G. K. Batchelor. *An Introduction to Fluid Dynamics*, chapter 4. Cambridge University Press, 1988.
- [2] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. unpublished, 2000.
- [3] Applied Numerical Algorithms Group. Incompressible Navier-Stokes software design. Technical report, NERSC Division, Lawrence Berkeley National Laboratory, 2002.
- [4] Dan Martin and Phil Colella. Incompressible Navier-Stokes with particles design document. Technical report, Applied Numerical Algorithms Group, Lawrence Berkeley Laboratory, 2003.
- [5] Dan Martin and Phil Colella. *Software Design for Particles in Incompressible Flow*. Applied Numerical Algorithms Group, Lawrence Berkeley Laboratory, 2003.